Algorithmic Ranking for Food Waste Apps

Milestone 0 Report

Mahinour Abdelgawad

Mennatallah Essam Zaher

Nadine El Garem

Ziad Khairaldin

CSCE2202 Analysis and Design of Algorithms

Fall 2025

Report Contents

Problem Description

In this project, we are studying how a food-waste app (similar to Too Good To Go) should decide which stores to show to each customer. Every morning, each bakery or store enters how many "surprise bags" they expect to have left by the end of the day. The app then shows a list of stores to customers, and customers choose from that list.

The current problem is that the app's display method is not fair and not efficient:

- Some popular stores are always shown first, even if they overestimate how many leftover bags they will have.
- These stores sometimes cancel customer reservations later in the day because they actually have fewer bags than expected.
- Customers become unhappy and may leave the platform.
- At the same time, less popular stores are barely shown to customers, so they end the day with unsold bags (more food waste).
- The platform loses money because unsold bags mean lost revenue, and cancelled reservations reduce trust.

So the system is not balanced.

Some stores oversell → angry customers.

Some stores are hidden → extra food waste.

Overall → lower revenue, lower fairness.

Our job is to design a ranking algorithm that chooses n stores to display to each customer and tries to fix these issues.

We need to understand:

- What is a good value of n? (How many stores should a customer see?)
- Which stores should be chosen for each customer?
- How do we balance between popular and less popular stores?
- How do we reduce reservation cancellations?
- How do we reduce food waste while still maximizing revenue?

Brainstorming, System Breakdown, and Preliminary Research

Your role is to design a ranking algorithm (no ML) that selects which n stores to display and what a good value for n could be. You want to minimize food waste while maximizing revenue and measure if your algorithm will be able to treat the stores more fairly.

To Do:

- Figure out how to decide on a good value for n
- How to select n stores to display each day?
- Measure if our algorithm will be able to treat the stores more fairly

Goal:

We need to figure out the best way to display n stores for each customer (it is different for each customer) such that we minimize food waste and maximize revenue. In other words: Some stores are popular but they overestimate their leftovers and some customers have their reservations cancelled, this will cause them to leave negative reviews on our app. On the other hand, some unpopular stores might have too many leftovers at the end with not enough customers because their store was not displayed on the page, this causes more food waste and less revenue.

What Data Do We Need?

1. *Data of the bakeries for the day*

   Example:

   We have M number of bakeries.

   For each bakery, we will collect and store:

   - Current bakery rating
   - Price of surprise bags
   - Estimated quantity of surprise bags at the start of the day
   - Actual quantity of surprise bags at the end of the day
   - Number of reservations cancelled
   - Previous data – all of these data points for all previous days (maybe not all days, but just the last x days but then how will we decide what days to consider?)

2. *Information from customers*

   We will need to collect data from all the users

   - What stores they previously bought bags from
   - What stores previously cancelled their reservations
   - What time they usually open the app/reserve a bag
   - Preferences?

Potential Algorithm Approaches

1) *Greedy:*

Implement an algorithm that predicts the number of N surprise bags at the end of the day(11 am) based on previous results and patterns, and set the N of surprise bags(8am) to a lower number than that(lets say K). At the same time, the app showcases the store's surprise bag to K number of people. As a result, we guarantee a local optimal solution where the customers and the bakeries have to get and sell their waste food most of the time.

2) *Brute Force(backtracking):*

Using the rating system of shops and the available number of N surprise bags for the store, the app will NOT assign the bakery to a customer if rating is high and N bags have been exceeded. Can be done through graph coloring where the program constantly checks if a bakery can be recommended to a customer

3) *Dynamic Programming*

Since iterating through the bakery's data again will be done frequently, utilizing DP can help with solving overlapping subproblems.

Research Points

- https://www.toogoodtogo.com/en-us/how-does-the-app-work → food waste app we can research

Constraints to Consider

1. We should consider location – only show stores that are near to the customer
2. We can only change what the customer sees. We can't control the stores .We can not influence their initial bag estimates, final true inventory, pricing decisions, and the actual contents of the surprise bags.
3. We should consider the limited information the app has as it receives only information about the store's estimated number of bags and the bag price – no information about the actual food items and their quantity

Questions to Ask

- Do we get frequent updates on the status of available surprise bags throughout the day or is it only at the beginning and end of the day?
- Does n/the list of stores change dynamically throughout the day or is it the same n/list throughout the day and it only changes the next day?
- Should the algorithm focus more on minimizing food waste or maximizing revenue if we cannot achieve both at the same time?
- Where should we get data from to work with?
- Is it possible for a customer to receive more than one surprise bag? Maybe from other bakeries too

Team Collaboration

There is a lot of tasks and research to be done for this project, including:

- Build the system simulation along with a visualized dashboard
- Generate data or scrape real data from similar businesses
- Implement the greedy baseline of top-n popular stores in order to be able to make
- comparisons
- Recommend and evaluate the ranking strategies (one strategy per student)

One suggestion for distributing the workload is:

- 2 students build the system simulation along with a visualized dashboard, and implement the greedy baseline of top-n popular stores in order to be able to make comparisons
- 2 students generate data or scrape real data from similar businesses
- Each student recommends and evaluates the ranking strategies